FAKTORZEHN

Faktor-IPS

Übungsunterlagen zur Schulung

Inhaltsverzeichnis

1 Einleitung

Die Übungen zur Einführungsschulung für Faktor-IPS richten sich sowohl an fachliche Anwender als auch an Java-Entwickler. Einige Aufgaben erfordern technische Kenntnisse und sind mit JAVA gekennzeichnet.

2 Übung zu Kapitel III.A: Projekt einrichten

2.1 Übung 1 JAVA

Legen Sie ein neues Maven-Projekt mit dem Namen „hausratmodell“ an:

- Deaktivieren Sie im Projektwizard die Schaltfläche „Create a simple project (skip archetype selection)“
- Filtern Sie die Liste der Archetypes nach „org.faktorips“
- Wählen Sie die gewünschte Faktor-IPS Version aus
- Group Id = org.faktorips.schulung
- Artifact Id = hausratmodell
- Package = org.faktorips.schulung.hausratmodell
- Im Properties Bereich wählen Sie folgende Eigenschaften aus
  - JavaVersion = 17
  - IPS-IsModelProject = true
  - IPS-IsProductDefinitionProject = false
  - IPS-SourceFolder = modell
- Nach der Bestätigung mit dem Button „Finish“ wird die Console im Eclipse gestartet und erwartet noch eine Property
  - IPS-RuntimeIdPrefix = hausrat.
  - Bestätigen Sie mit „Y“

Alternativ kann das Projekt auch über die Kommandozeile angelegt werden. Geben Sie dazu folgenden maven Befehl in eine Konsole ein:

```
mvn archetype:generate -DarchetypeGroupId=org.faktorips \
-DarchetypeArtifactId=faktorips-maven-archetype \
-DarchetypeVersion=23.6.1.release
```

Dabei werden die wichtigsten Einstellungen Interaktiv abgefragt. Sollte einer der vorbelegten Einstellungen verändern werden müßen, kann die Eingabe mit „N“ erfolgen. Dadurch werden alle wichtigen Eigenschaften eines Faktor-IPS abgefragt.

Sie sollten jetzt im Project- oder Package-Explorer das Faktor-IPS Projekt „hausratmodell“ sehen.

Öffnen Sie den Projekt-Eigenschaften-Dialog:

Kontextmenü auf dem Projekt „hausratmodell“

→ Properties

→ links „Faktor-IPS Code Generator“ auswählen

→ rechts „Builder-Klassen generieren“ auf „None“ stellen und

→ „Generator Language“ zu „de“

3 Übung zu Kapitel III.B: Erste Vertragsklasse anlegen

3.1 Übung 1

Öffnen sie den Faktor-IPS Modellexplorer (Der Modellexplorer befindet sich neben dem Package-Explorer. Ansonsten über Menü -> Window -> Show View -> ganz unten "Other" -> "Modell-Explorer" ins Suchfeld eingeben und auswählen.)

Legen Sie im Sourceverzeichnis "modell" das Faktor-IPS Package "hausrat" an. (Im ModellExplorer das Kontextmenü des Projekts HausratModell aufrufen und "Neu" auswählen.)

Legen Sie eine Vertragsteilklasse "HausratVertrag" in dem Package "hausrat" an. (Im Dialog die Option "Instanzen sind durch Produktbausteine konfigurierbar" NICHT anhaken!)

Sehen Sie sich die generierte Javaklasse HausratVertrag im Java Package Explorer an.

4 Übung zu Kapitel III.B.1: Attribute auf Vertragsseite

4.1 Übung 1

Legen Sie die folgenden Attribute in der Klasse HausratVertrag im ModellExplorer an.

Name : Datentyp	***Beschreibung, Bemerkung***
zahlweise : Integer	Die Zahlweise des Vertrages. Die erlaubten Werte sind 1, 2, 4 und 12.
plz : String	Postleitzahl des versicherten Hausrats. Maximal 5 Zeichen.
/tarifzone : String	Die Tarifzone ergibt sich aus der Postleitzahl und ist maßgeblich für den zu zahlenden Beitrag. => Achten sie also bei der Eingabe darauf, die Eigenschaft „Typ des Attributes“ auf „abgeleitet (Berechnung bei jedem Aufruf...)“ zu setzen!
wohnflaeche : Integer	Die Wohnfläche des versicherten Hausrats in Quadratmetern. Der erlaubte Wertebereich ist min=0 und max=unbeschränkt. Hierzu muss das Feld Maximum leer bleiben.
/vorschlagVersSumme : Money	Vorschlag für die Versicherungssumme. Wird auf Basis der Wohnfläche bestimmt. => Achten sie bei der Eingabe darauf, die Eigenschaft „Typ des Attributes“ auf „abgeleitet (Berechnung bei jedem Aufruf...)“ zu setzen!
versSumme : Money	Die Versicherungssumme. Der erlaubte Wertebereich ist min=0 EUR und max=unbeschränkt.

4.2 Übung 2 JAVA

Probieren Sie die Wirkungsweise der Annotationen @generated, @generated NOT im generierten Code aus.

4.3 Übung 3 JAVA

Implementieren Sie die Ermittlung der Tarifzone. Geben Sie immer Tarifzone „I“ zurück. (Später werden wir die Tarifzone anhand der Postleitzahl aus einer Tarifzonentabelle ermitteln.)

Implementieren Sie die Ermittlung des Vorschlagswertes für die Versicherungssumme. Der Vorschlag ergibt sich durch Wohnfläche * 650 €. (Später werden wir die 650 € produktseitig konfigurierbar machen.)

4.4 Übung 4 JAVA *(Optional)*

Erstellen Sie einen Junit-Testfall HausratVertragTest. Schreiben Sie Testfälle für die beiden Methoden *getTarifzone()* und *getVorschlagVersSumme()*.

TIP: Nutzen Sie MoreUnit (https://marketplace.eclipse.org/content/moreunit) um die Testklasse und –methoden zu erstellen (Strg+J).

In der pom.xml können Sie dazu in den jeweiligen Abschnitten folgendes ergänzen:

```
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.hamcrest</groupId>
            <artifactId>hamcrest</artifactId>
            <version>2.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
[…]
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.junit</groupId>
                <artifactId>junit-bom</artifactId>
                <version>5.8.2</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
[…]
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.2</version>
            </plugin>
        </plugins>
    </build>
```

5 Übung zu III.B.2: Beziehungen auf Vertragsseite

Legen Sie den Vertragsteil-Typ "HausratGrunddeckung" im Package "hausrat" an.

Legen Sie die Beziehung vom Beziehungstyp "Komposition" zwischen HausratVertrag und HausratGrunddeckung mit der Kardinalität [1..1] an.

Starten sie dazu im Editor für den HausratVertrag in der Section "Beziehungen". Folgen sie den Anweisungen des Dialogs. Auf der Dialog-Seite "Inverse Beziehung" wählen sie "Neue inverse Beziehung" aus. Geben sie dann die Kardinalität ist [1..1] an.

Schauen Sie sich den generierten Sourcecode an.

6 Übung zu Kapitel III.C.1: Attribute auf Produktseite

6.1 Übung 1 JAVA

Legen Sie wie beim Modell beschrieben ein neues Maven-Projekt „Hausratprodukte“ mit den folgenden Eigenschaften an:

- Group Id = org.faktorips.schulung
- Artifact Id = hausratprodukte
- Package = org.faktorips.schulung.hausratprodukte
- Im Properties Bereich wählen Sie folgende Eigenschaften aus
  - JavaVersion = 17
  - IPS-IsModelProject = false
  - IPS-IsProductDefinitionProject = true
  - IPS-SourceFolder = produktdaten
  - IPS-IsGroovySupport = true
- In der Interaktiven „Console“
  - IPS-RuntimeIdPrefix = hausrat.
  - Bestätigen Sie mit „Y“

Öffnen Sie den Projekt-Eigenschaften-Dialog:

Kontextmenü auf dem Projekt „Hausratmodell“

→ Properties

→ links „Faktor-IPS Code Generator“ auswählen

→ rechts „Formel-Kompilierung“ auf „XML“ stellen

Fügen Sie eine Maven-Dependency in der pom.xml Datei hinzu, um das Modellprojekt miteinzubeziehen:

```
<dependency>
  <groupId>org.faktorips.schulung</groupId>
  <artifactId>hausratmodell</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</dependency>
```

6.2 Übung 2

Legen Sie den Produktbaustein-Typ „HausratProdukt“ im Projekt „Hausratmodell“ an und tragen dabei „HausratVertrag“ als Vertragsteiltyp ein.

Definieren Sie das Attribut „produktname“ mit Datentyp String.

Schauen Sie sich den generierten Sourcecode an.

Wechseln Sie in die Produktdefinitionsperspektive.

Legen Sie im Projekt „Hausratprodukte“ im Sourceverzeichnis „produktdaten“ ein neues Package „produkte_jjjj“ an, wobei jjjj für das aktuelle Jahr steht.

Legen Sie die beiden Produktbausteine „HR-Kompakt“ und „HR-Optimal“ vom Produktbaustein-Typ HausratProdukt an. Geben sie dazu im Dialog den Namen des Produktbausteins und als „Gültig Ab“ Datum den 1.1. dieses Jahres ein. Geben Sie anschließend im Editor den jeweiligen Produktnamen „Hausrat Kompakt“ und „Hausrat Optimal“ ein.

6.3 Übung 3

Wechseln Sie zurück in die Java-Perspektive.

Dokumentieren Sie die Klasse HausratProdukt und das Attribut „produktname“.

Legen Sie ein weiteres Attribut (beispielsweise Kurzbezeichnung) an der Klasse HausratProdukt an. Wechseln Sie zurück in die Produktdefinitionsperspektive und tragen Sie Werte für das neue Attribut in den beiden Produkten Kompakt und Optimal ein.

7 Übung zu Kapitel III.C.2: Beziehungen auf Produktseite

Legen Sie den Produktbaustein-Typ „HausratGrunddeckungstyp“ an und berücksichtigen Sie, dass dieser den Vertragsteil-Typ „HausratGrunddeckung“ konfiguriert.

Legen Sie die Beziehung zwischen „HausratProdukt“ und „HausratGrunddeckungstyp“ an. (Jedes Hausratprodukt soll genau einen Grundeckungstypen enthalten). Starten Sie dazu im Editor für das Hausratprodukt in der Sektion „Beziehungen“ über den Button „Neu…“ den Wizard zum Anlegen einer Beziehung. Im Dialog „Beziehung bearbeiten“ bitte die Option „Kann in Generationen verändert werden“ NICHT anhaken.

Wechseln Sie in die Produktdefinitionsperspektive. Legen Sie jetzt die Deckungsbausteine „HRD-Optimal“ und „HRD-Kompakt“ an und fügen Sie die Grunddeckungsbausteine dem jeweiligen Produkt hinzu. Setzen Sie dabei die "Art der Beziehung“ auf „Obligatorisch“.

8 Übung zu Kapitel III.C.3: Konfigurierbare Vertragsattribute

Markieren Sie die folgenden Attribute im Vertragsteil-Typ „HausratVertrag“ als konfigurierbar (Dialog „Attribut bearbeiten“, Option „Der Vorbelegungswert und die Menge der erlaubten Werte sind in der Produktkonfiguration definiert“, Option „Kann in Anpassungsstufen verändert werden“ NICHT anhaken) und legen sie die Eigenschaft „Typ der Wertemenge“ auf dem Reiter „Vorbelegung und Werte“ des „Attribut bearbeiten“-Dialogs fest:

- zahlweise, Aufzählung
- wohnflaeche, Bereich
- versSumme, Bereich

Geben Sie die Daten für die beiden Produkte gemäß der folgenden Tabelle ein:

Konfigurationsmöglichkeit	***HR-Kompakt***	***HR-Optimal***
Erlaubte Zahlweisen	monatlich, jährlich	monatlich, vierteljährlich, halbjährlich, jährlich
Defaultwert Zahlweise	monatlich	jährlich
Erlaubter Bereich Wohnfläche	0-100 qm	0-200 qm
Defaultwert Wohnfläche	<keine Vorbelegung>	<keine Vorbelegung>
Erlaubter Bereich VersSumme	10Tsd – 2Mio Euro	10Tsd – 5Mio Euro
Defaultwert VersSumme	<keine Vorbelegung>	<keine Vorbelegung>

9 Übung zum Kapitel III.C.4: Zugriff auf Produktdaten zur Laufzeit

9.1 Übung 1 JAVA

Legen Sie im Produktdatenprojekt eine JUnit-Testklasse an. Binden Sie dazu die JUnit-Library wie oben in Übung 4.4 beschrieben als Maven-Dependency ein. Zusätzlich wird eine Dependency zu einer JAXB-API verwendet, da diese seit Java 11 nicht mehr enthalten ist:

```
<dependency>
  <groupId>jakarta.xml.bind</groupId>
  <artifactId>jakarta.xml.bind-api</artifactId>
  <version>4.0.0</version>
</dependency>
```

Hinweis: Sollte die JUnit Klasse nicht als Testfall erkannt werden, überprüfen Sie, ob in den Projekteinstellungen->Java Compiler das richtige JDK hinterlegt ist (min. Java8)!

Geben sie die folgenden Informationen über das Produkt Kompakt (in einer test()-Methode) auf der Konsole aus:

- Produktname
- Defaultwert für die Zahlweise
- Erlaubte Zahlweisen
- Erlaubter Wertebereich für die Versicherungssumme
- Erlaubter Wertebereich für die Wohnfläche

Für den Zugriff auf das Repository verwenden Sie folgenden Setup-Code.

```
private static IRuntimeRepository repository;

@BeforeAll
static void setUp() {
    // Repository erzeugen
    repository = ClassloaderRuntimeRepository
        .create("org/faktorips/schulung/hausratprodukte/internal/faktorips-
repository-toc.xml");
}
```

Das Produkt erhält man durch den Aufruf der Methode getProductComponent(String id) des RuntimeRepository. Die ID eines Bausteins finden Sie auf der zweiten Seite im Baustein-Editor.

9.2 Übung 2

Definieren Sie das das Attribut „vorschlagVersSummeProQm“ am Produktbaustein-Typ HausratProdukt.

Ergänzen Sie die Produktbausteine um die Werte aus der folgenden Tabelle.

Produktbaustein	*vorschlagVersSummeProQm*
HR-Optimal	900 €
HR-Kompakt	600 €

9.3 Übung 3 JAVA

Schreiben Sie dazu einen Testfall für die Methode getVorschlagVersSumme () für das Produkt Kompakt.

Implementieren Sie dann die Berechnung des Vorschlags für die Versicherungssumme in der Java-Klasse HausratVertrag. (Lösung im Sourcecode-Ausschnitt 1 im Anhang).

10 Übungen zu Kapitel III.D.1: Grundlagen der Verwendung von Tabellen

10.1 Übung 1

Legen Sie die Tabellenstruktur „Tarifzonentabelle“ im Projekt „Hausratmodell“. Wählen sie als Tabellentyp „SINGLE_CONTENT“ aus. Definieren Sie die Spalten „plzVon“, „plzBis“ und „Tarifzone“ vom Typ String und zusätzlichen einen Bereich für plzVon und plzBis. Legen Sie den eindeutigen Schlüssel auf diesen Bereich an.

Wechseln Sie in die Produktdefinitionsperspektive. Legen Sie den zugehörigen Tabelleninhalt „Tarifzonentabelle“ im Projekt „Hausratprodukte“ an.

Plz-Von	*Plz-bis*	*Tarifzone*
17235	17237	II
45525	45549	III
59174	59199	IV
47051	47279	V
63065	63075	VI

10.2 Übung 2 JAVA

Schreiben Sie einen JUnit-Testmethode für die Methode HausratVertrag.getTarifzone und implementieren Sie die Methode unter Verwendung der Tarifzonentabelle. Die Methode soll für Postleitzahlen, die nicht in einen Bereich der Tarifzonentabelle fallen, nach wie vor „I“ zurückgeben. Den Code finden sie im Anhang im Code-Ausschnitt 2.

Hinweis: Die Tabelle kann über das Runtime-Repository geladen werden:

```
Tarifzonentabelle tabelle = getProductComponent().getRepository()
          .getTable(Tarifzonentabelle.class);
```

11 Übungen zu Kapitel III.D.2: Zusammenhang Bausteine-Tabellen

11.1 Übung 1

Legen Sie die Tabellenstruktur „TariftabelleHausrat“ im Projekt „Hausratmodell“ an. Stellen Sie den Tabellentyp auf „MULTIPLE_CONTENTS“. Die Tarifzone ist vom Typ String, der Beitragssatz vom Typ Decimal. Als eindeutigen Schlüssel verwenden sie die Tarifzonenspalte. Definieren Sie im Modellprojekt, dass die Klasse HausratGrundeckungstyp die TariftabelleHausrat als „Tariftabelle“ verwendet und ein Tabelleninhalt erforderlich ist.

Wechseln Sie in die Produktdefinitionsperspektive und legen Sie im Projekt Hausratprodukte die Tariftabellen für die beiden Hausratprodukte an und ordnen Sie die Tabellen der jeweiligen Grunddeckung zu.

Tariftabelle-Optimal

Tarifzone	*Beitragssatz pro 1000 Euro*
I	0.8
II	1.0
III	1.2
IV	1.4
V	1.6
VI	1.8

Tariftabelle-Kompakt

Tarifzone	*Beitragssatz pro 1000 Euro*
I	0.6
II	0.8
III	1.0
IV	1.2
V	1.4
VI	1.6

11.2 Übung 2 JAVA

Legen Sie an der Klasse HausratGrunddeckung ein abgeleitetes (cached) Attribut „jahresbasisbeitrag“ vom Typ Money an. Der Jahresbasisbeitrag ist der jährlich zu zahlende Beitrag ohne Zuschläge und Nachlässe, ohne Ratenzahlungszuschlag und ohne Versicherungssteuer.

Definieren Sie eine Methode void und ohne Paramter „berechneJahresbasisbeitrag“ in der Modellklasse HausratGrundeckung (nicht direkt im Java-Code) zur Berechnung des Jahresbasisbeitrags.

Schreiben Sie einen JUnit-Test für die Methode und implementieren Sie diese. Verwenden Sie hierzu die im Baustein zugeordnete Tariftabelle.

Die Berechnungsvorschrift ist wie folgt:

- Ermittlung des Beitragssatzes aus der Tariftabelle
- Division der Versicherungssumme durch 1000 und Multiplikation mit dem Beitragssatz

Hinweis:

Sie müssen wie in den vorangegangenen Testfällen zuerst einen HausratVertrag erzeugen und dessen Attribute setzen. Zusätzlich müssen Sie nun eine HausratGrunddeckung erzeugen und zum Vertrag hinzufügen. Dies geht wie folgt:

```
// Grunddeckungstyp holen, der dem Produkt zugeordnet ist
HausratGrunddeckungstyp deckungstyp = produkt.getHausratGrunddeckungstyp();

// Grunddeckung erzeugen und zum Vertag hinzufügen
HausratGrunddeckung deckung = vertrag.newHausratGrunddeckung(deckungstyp);
```

12 Übungen zu Kapitel III.E: Aufzählungen

12.1 Übung 1

Legen Sie den Aufzählungstypen „Zahlweise“ im Projekt „Hausratmodell“ mit den folgenden Werten an, wobei die Zahlungen pro Jahr vom Datentyp int als ID dienen.

Zahlungen pro Jahr	(Fachliche) Bezeichnung
12	Monatlich
4	Quartalsweise
2	Halbjährlich
1	Jährlich
0	Einmalzahlung

12.2 Übung 2

Ändern Sie den Datentyp des Attributs „zahlweise“ am HausratVertrag auf den soeben angelegten Aufzählungstyp.

Kontrollieren Sie, ob in den Produkten Hausrat Kompakt und Optimal nun die Bezeichnung der Zahlweisen zu sehen sind, anstatt der ID.

12.3 Übung 3 JAVA *(Optional)*

Legen Sie an der HausratGrunddeckung das berechnete (bei jedem Aufruf) Attribut „beitragGemaessZahlweise“ an.

Testen und implementieren sie die Berechnung des Beitrages gemäß Zahlweise indem der Jahresbasisbeitrag geteilt durch die Anzahl der Zahlungen pro Jahr aus der gewählten Zahlweise (ID) zurückgegeben wird.

Für Einmalzahlung kann Money.NULL zurückgegeben werden.

Beachten Sie auch, dass weder Zahlweise noch Jahresbasisbeitrag schon gesetzt sein müssen.

12.4 Übung 4

Legen Sie den Aufzählungstypen „Risikoklasse“ im Projekt „Hausratmodell“ an. Im Dialog „Erstelle Aufzählungstyp“ die Option „Erweiterbar durch separaten Inhalt“ anhaken. Als Name für die Aufzählung, die die tatsächlichen Werte enthält, einfach „Risikoklasse“ angeben (also nicht die Voreinstellung „hausrat.Risikoklasse“ übernehmen).

Im Projekt „Hausratprodukte“ über das Kontextmenü des Modellexplorers → Aufräumen → Fehlende Aufzählungsinhalte anlegen → ok. Nun in die Produktdefinitionsperspektive wechseln und die konkreten Risikoklassen eintragen:

ID	Name
10	Ständig bewohntes Einfamilienhaus
20	Ständig bewohntes Mehrfamilienhaus
30	Ständig bewohntes Ferienhaus
40	Zweitwohnung

12.5 Übung 5 JAVA

Eine neue Test-Methode anlegen und sich Risikoklasse ausgeben lassen.

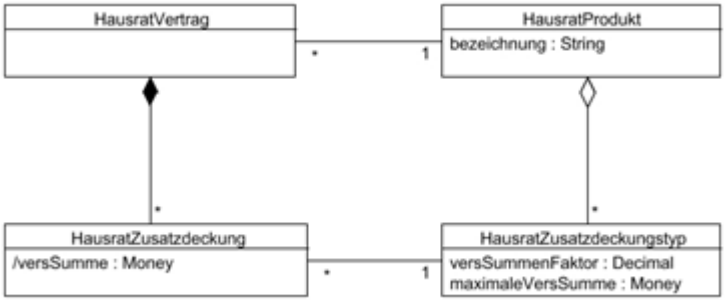
Die Werte der Aufzählung erhält man über das Runtime Repository:

```
List<Risikoklasse> risikoklassen = repository.getEnumValues(Risikoklasse.class);
```

13 Übungen zu Kapitel III.F: Verwendung von Formeln

13.1 Übung 1

Erweitern Sie das Hausratmodell um die „HausratZusatzdeckung“ und den „HausratZusatzdeckungstyp“ gemäß dem folgenden Diagramm (versSumme berechnet bei jedem Aufruf):



Achten Sie darauf beim Erzeugen der Beziehung zwischen HausratVertrag und HausratZusatzdeckung auch die Rückbeziehung anzulegen.

Legen Sie die folgenden Zusatzdeckungen an und ordnen Sie sie den Produkten zu:

	Fahrraddiebstahl 2023-01	*Überspannung 2023-01*
VersSummenFaktor	0,01	0,05
MaximaleVersSumme	5000EUR	<null>

13.2 Übung 2 JAVA

Testen und implementieren Sie die Berechnung der Versicherungssumme der Zusatzdeckung in der Methode getVersSumme().

13.3 Übung 3

Legen Sie an der HausratZusatzdeckung das Attribut „jahresbasisbeitrag“ (berechnet bei jedem Aufruf) an und wählen „Der Wert des Attributes wird durch eine Methode des Typs ‚HausratZusatzdeckungstyp‘ berechnet“ an. Legen Sie die Berechnungsmethode an indem Sie auf den angebotenen Link klicken.

Definieren Sie nun den Jahresbasisbeitrag in den Bausteinen wie folgt:

- Fahrraddiebstahl: 10% der Versicherungssumme der Deckung
- Überspannung: 30EUR + 3% der Versicherungssumme der Deckung

Passen Sie die Faktor-IPS-Codegenerator-Einstellungen an:

- Formel-Kompilierung: XML

Stellen Sie sicher, dass in der pom.xml Die Abhängigkeit zu faktorips-runtime-groovy eingetragen ist:

```
<dependency>
    <groupId>org.faktorips</groupId>
    <artifactId>faktorips-runtime-groovy</artifactId>
    <version>>${faktorips.version}</version>
</dependency>
```

13.4 Übung 4 JAVA

Implementieren und testen Sie die Methode HausratZusatzdeckung#getJahresbasisbeitrag() mit einem Aufruf der Methode am HausratZusatzdeckungstyp.

Zur Ausführung von Formeln aus XML wird im Setup des Tests dem ClassloaderRuntimeRepository die GroovyFormulaEvaluatorFactory gesetzt:

```
repository.setFormulaEvaluatorFactory(new GroovyFormulaEvaluatorFactory());
```

Diese Methode ist nicht im Interface IRuntimeRepository verfügbar.

14 Übungen zu Kapitel III.G: Plausibilisierungen

14.1 Übung 1

Legen Sie im Modell folgende Regeln an:

pruefeWohnflaeche (HausratVertrag)	Prüfen Sie, dass die Wohnfläche im konfigurierten Bereich liegt. Legen Sie die Regel über den Dialog für das Attribut wohnflaeche an und geben Sie einen sprechenden Fehlertext mit dem eingegebenen Wert und dem erlaubten Bereich an.
pruefePlz (HausratVertrag)	Prüfen Sie, dass die Postleitzahl aus 5 Ziffern besteht. Die Message sollte das Attribut enthalten und den angegebenen Wert im Fehlertext berücksichtigen.

14.2 Übung 2 JAVA

Fügen Sie in der pom.xml von Hausratprodukte die Abhängigkeit zu faktorips-testsupport ein:

```
<dependency>
    <groupId>org.faktorips</groupId>
    <artifactId>faktorips-testsupport</artifactId>
    <version>${faktorips.version}</version>
</dependency>
```

In der Java Klasse `org.faktorips.testsupport.IpsMatchers` befinden sich nützliche Hamcrest Matchers für MessageLists und Message.

Legen Sie eine JUnit-Testmethode an, der am HausratVertrag die *validate()*-Methode aufruft. Implementieren Sie die Testkonstellation so, dass die Prüfungen fehlschlagen.

Überprüfen Sie in der zurückgegebenen MessageList folgendes:

- Sie enthält die jeweilige Fehlermeldung.
- Prüfen Sie, dass alle konfigurierten Eigenschaften einer Regel (z.B: Code, geprüfte Attribute und Fehlermeldungen vorliegen.

Legen Sie eine weitere JUnit-Testmethode an, bei der *validate()* am HausratVertrag keine Fehlermeldung zurückgibt.

Implementieren Sie nun die Prüflogik. Beachten Sie dabei, dass die Parameter für die Fehlermeldungen gefüllt werden müssen.

14.3 Übung 3

Aktivieren Sie für die Versicherungssumme die generische Validierung und tragen „Versicherungssumme“ als Label ein.

14.4 Übung 4 JAVA

Testen Sie auch die generische Validierung, in einem JUnit-Test. Testen Sie, dass das Label sowie die Grenzen des Bereichs in der Fehlermeldung enthalten sind.

15 Übung zu Kapitel IV.A: Customizing der Produktdefinitionsperspektive

15.1 Übung 1

Kategorien

- Legen Sie eine Kategorie Versicherungssumme im Zusatzdeckungstypen an.
- Verschieben Sie die entsprechenden Attribute in diese Kategorie
- Legen Sie eine Kategorie Beitragsberechnung in den Deckungstypen an.
- Verschieben Sie die Tabellen / Formelsignaturen in diese neue Kategorie

15.2 Übung 2

Icons

- Weisen Sie den Produktbaustein-Typen Icons zu.

15.3 Übung 3

Labels

- Geben Sie Labels für das HausratProdukt, den Grunddeckungstypen und den Zusatzdeckungstypen ein.
- Geben Sie Labels für die Attribute Versicherungssumme, Maximale Versicherungssumme und den Versicherungssummenfaktor ein.

Schließen Sie alle Editoren und wechseln in die Produktdefinitionsperspektive, um sich das Ergebnis anzusehen.

16 Übung zu Kapitel IV.B: Kopieren von Produkten

Wechseln Sie in die Produktdefinitionsperspektive. Selektieren Sie das bestehende Produkt HR-Optimal im Produktstrukturexplorer

Wählen Sie über das Kontextmenü „Neue Version erzeugen…“

Legen Sie eine neue Folgegeneration zum Beginn des nächsten Jahres an und kopieren alle Bausteine.

Erzeugen Sie eine Wasserbettendeckung durch Kopieren der bestehenden Fahrraddiebstahldeckung und fügen Sie die neue Deckung zum Produkt HR-Optimal hinzu.

17 Anhang Lösungen Programmieraufgaben

17.1 Übung 4.4: Tests für Ermittlung Tarifzone und Vorschlag Versicherungssumme

17.1.1 Klasse: HausratVertrag

```
public Money getVorschlagVersSumme() {
  // begin-user-code
  return Money.euro(650).multiply(getWohnflaeche());
   // end-user-code
}

public String getTarifzone() {
    // begin-user-code
    return "I";
    // end-user-code
}
```

17.1.2 Testklasse HausratVertragTest

```
@Test
void testGetTarifzone() {
    HausratVertrag hausratVertrag = new HausratVertrag();

    assertThat(hausratVertrag.getTarifzone(), is("I"));
}
@Test
void testGetVorschlagVersSumme() {
    HausratVertrag hausratVertrag = new HausratVertrag();
    hausratVertrag.setWohnflaeche(100);

    assertThat(hausratVertrag.getVorschlagVersSumme(), is(Money.euro(65_000)));
}
@Test
void testGetVorschlagVersSumme_WohnflaecheNichtGesetzt() {
    HausratVertrag hausratVertrag = new HausratVertrag();

    assertThat(hausratVertrag.getVorschlagVersSumme(), is(Money.NULL));
}
```

17.2 Übung 9.1: Zugriff auf Produktdaten

17.2.1 Testklasse HausratProduktTest

```
private static IRuntimeRepository repository;
private static HausratProdukt hrKompakt;

@BeforeAll
static void setUp() {
    // Repository erzeugen
    repository = ClassloaderRuntimeRepository
            .create("org/faktorips/schulung/hausratprodukte/internal/faktorips-repository-toc.xml");
    // Referenz auf das Kompaktprodukt aus dem Repository holen
    hrKompakt = (HausratProdukt) repository.getProductComponent("hausrat.HR-Kompakt 2023-01");
}

@Test
void testProduktdatenLesen() {
    System.out.println("Produktname:                        " + hrKompakt.getProduktname());
    System.out.println("Zahlweise(Default):                 " + hrKompakt.getDefaultValueZahlweise());
    System.out.println("Zahlweise(erlaubte Werte):          "
            + hrKompakt.getAllowedValuesForZahlweise());
    System.out.println("Versicherungssumme(erlaubte Werte): " + hrKompakt. getAllowedValuesForVersSumme
());
    System.out.println("Wohnfläche (erlaubte Werte):        "
            + hrKompakt.getAllowedValuesForWohnflaeche());
}
```

17.3 Übung 9.3: Ermittlung Vorschlag Versicherungssumme

17.3.1 Klasse: HausratVertrag

```
public Money getVorschlagVersSumme() {
    // begin-user-code
    HausratProdukt produkt = getHausratProdukt();
    if (produkt == null) {
        return Money.NULL;
    }
    return produkt.getVorschlagVersSummeProQm().multiply(getWohnflaeche());
    // end-user-code
}
```

17.3.2 Testklasse HausratProduktTest

```
@Test
void testGetVorschlagVersSumme() {
    // Erzeugen eines Hausratvertrags mit der Factorymethode des Produktes
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(100);

    assertThat(hausratVertrag.getVorschlagVersSumme(), is(Money.euro(60_000)));
}

@Test
void testGetVorschlagVersSumme_KeineWohnflaeche() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();

    assertThat(hausratVertrag.getVorschlagVersSumme(), is(Money.NULL));
}

@Test
void testGetVorschlagVersSumme_KeinProdukt() {
    HausratVertrag hausratVertrag = new HausratVertrag();

    assertThat(hausratVertrag.getVorschlagVersSumme(), is(Money.NULL));
}
```

17.4 Übung 10.2: Ermittlung Tarifzone

17.4.1 Klasse: HausratVertrag

```
public String getTarifzone() {
    // begin-user-code
    IProductComponent productComponent = getProductComponent();
    if (productComponent != null) {
        Tarifzonentabelle tabelle = productComponent.getRepository().getTable(Tarifzonentabelle.class);
        TarifzonentabelleRow tarifzonentabelleRow = tabelle.findRow(plz);
        if (tarifzonentabelleRow != null) {
            return tarifzonentabelleRow.getTarifzone();
        }
    }
    return "I";
    // end-user-code
}
```

17.4.2 Testklasse HausratProduktTest

```
@Test
void testGetTarifzone() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");

    assertThat(hausratVertrag.getTarifzone(), is("V"));
}
```

```
@Test
void testGetTarifzone_KeinePLZ() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();

    assertThat(hausratVertrag.getTarifzone(), is("I"));
}

@Test
void testGetTarifzone_PLZnichtInTabelle() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("80639");

    assertThat(hausratVertrag.getTarifzone(), is("I"));
}

@Test
void testGetTarifzone_KeinProdukt() {
    HausratVertrag hausratVertrag = new HausratVertrag();

    assertThat(hausratVertrag.getTarifzone(), is("I"));
}
```

17.5 Übung 11.2: Ermittlung Jahresbasisbeitrag Grunddeckung

17.5.1 Klasse: HausratGrunddeckung

```
/**
 * @generated NOT
 */
public void berechneJahresbasisbeitrag() {
    jahresbasisbeitrag = Money.NULL;
    String tarifzone = hausratVertrag.getTarifzone();
    TariftabelleHausratRow tariftabelleHausratRow = getTariftabelle().findRow(tarifzone);
    if (tariftabelleHausratRow != null) {
        Decimal beitragssatz = tariftabelleHausratRow.getBeitragssatz();
        Money versSumme = hausratVertrag.getVersSumme();
        jahresbasisbeitrag = versSumme.divide(1000, RoundingMode.HALF_UP)
                .multiply(beitragssatz, RoundingMode.HALF_UP);
    }
}
```

17.5.2 Testklasse HausratProduktTest

```
@Test
void testBerechneJahresbasisbeitrag() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    hausratVertrag.setVersSumme(Money.euro(60_000));
    // Grunddeckungstyp holen, der dem Produkt zugeordnet ist.
    HausratGrunddeckungstyp hausratGrunddeckungstyp =
        hrKompakt.getHausratGrunddeckungstyp();
    // Grunddeckung erzeugen und zum Vertag hinzufügen
    HausratGrunddeckung hausratGrunddeckung =
hausratVertrag.newHausratGrunddeckung(hausratGrunddeckungstyp);

    hausratGrunddeckung.berechneJahresbasisbeitrag();

    assertThat(hausratGrunddeckung.getJahresbasisbeitrag(), is(Money.euro(84)));
}

@Test
void testBerechneJahresbasisbeitrag_KeinePLZ() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    // keine PLZ -> Tarifzone I
    hausratVertrag.setVersSumme(Money.euro(60_000));
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getJahresbasisbeitrag(), is(Money.euro(36)));
}
```

```
@Test
void testBerechneJahresbasisbeitrag_KeineVersSumme() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getJahresbasisbeitrag(), is(Money.NULL));
}
```

17.6 Übung 12.3: Ermittlung Beitrag gemäß Zahlweise

17.6.1 Klasse HausratGrunddeckung

```
public Money getBeitragGemaessZahlweise() {
    // begin-user-code
    Zahlweise zahlweise = getHausratVertrag().getZahlweise();
    if (zahlweise == null || Zahlweise.EINMALZAHLUNG == zahlweise
            || jahresbasisbeitrag.isNull())
        return Money.NULL;
    return jahresbasisbeitrag
            .divide(zahlweise.getZahlungenProJahr(), RoundingMode.HALF_UP);
    // end-user-code
}
```

17.6.2 Testklasse HausratProduktTest

```
@Test
void testBerechneBeitragGemaessZahlweise() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    hausratVertrag.setVersSumme(Money.euro(60_000));
    hausratVertrag.setZahlweise(Zahlweise.HALBJAEHRLICH);
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getBeitragGemaessZahlweise(),
            is(Money.euro(42)));
}

@Test
void testBerechneBeitragGemaessZahlweise_KeinJahresbasisbeitrag() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    hausratVertrag.setVersSumme(Money.euro(60_000));
    hausratVertrag.setZahlweise(Zahlweise.HALBJAEHRLICH);
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    // keine Berechnung
    // hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getBeitragGemaessZahlweise(), is(Money.NULL));
}

@Test
void testBerechneBeitragGemaessZahlweise_KeineZahlweise() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    hausratVertrag.setVersSumme(Money.euro(60_000));
    hausratVertrag.setZahlweise(null);
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getBeitragGemaessZahlweise(), is(Money.NULL));
}
```

```
@Test
void testBerechneBeitragGemaessZahlweise_Einmalzahlung() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setPlz("47110");
    hausratVertrag.setVersSumme(Money.euro(60_000));
    hausratVertrag.setZahlweise(Zahlweise.EINMALZAHLUNG);
    HausratGrunddeckung hausratGrunddeckung = hausratVertrag
            .newHausratGrunddeckung(hrKompakt.getHausratGrunddeckungstyp());

    hausratGrunddeckung.berechneJahresbasisbeitrag();
    assertThat(hausratGrunddeckung.getBeitragGemaessZahlweise(), is(Money.NULL));
}
```

17.7 Übung 12.5: Zugriff auf erweiterbare Aufzählungen

17.7.1 Testklasse HausratProduktTest

```
@Test
void testErweiterbareAufzählung() {
    repository.getEnumValues(Risikoklasse.class).forEach(System.out::println);
}
```

17.8 Übung 13.2: Ermittlung Versicherungssumme Zusatzdeckung

17.8.1 Klasse: HausratZusatzdeckung

```
public Money getVersSumme() {
    // begin-user-code
    HausratZusatzdeckungstyp zusatzdeckungstyp = getHausratZusatzdeckungstyp();
    if (zusatzdeckungstyp == null) {
        return Money.NULL;
    }
    Decimal faktor = zusatzdeckungstyp.getVersSummenFaktor();
    Money vsVertrag = getHausratVertrag().getVersSumme();
    if (vsVertrag.isNull()) {
        return Money.NULL;
    }
    Money vs = vsVertrag.multiply(faktor, RoundingMode.HALF_UP);
    Money maxVs = zusatzdeckungstyp.getMaximaleVersSumme();
    if (vs.greaterThan(maxVs)) {
        return maxVs;
    }
    return vs;
    // end-user-code
}
```

17.8.2 Testklasse HausratProduktTest

```
@Test
void testGetVerSumme_Zusatzdeckung_Fahrraddiebstahl() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setVersSumme(Money.euro(60_000));
    HausratZusatzdeckungstyp fahrraddiebstahlTyp = (HausratZusatzdeckungstyp) repository
            .getProductComponent("hausrat.Fahrraddiebstahl 2023-01");
    HausratZusatzdeckung fahrraddiebstahl = hausratVertrag.newHausratZusatzdeckung(fahrraddiebstahlTyp);

    assertThat(fahrraddiebstahl.getVersSumme(), is(Money.euro(600)));
}

@Test
void testGetVerSumme_Zusatzdeckung_Fahrraddiebstahl_Maximiert() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setVersSumme(Money.euro(600_000));
    HausratZusatzdeckungstyp fahrraddiebstahlTyp = (HausratZusatzdeckungstyp) repository
            .getProductComponent("hausrat.Fahrraddiebstahl 2023-01");
    HausratZusatzdeckung fahrraddiebstahl = hausratVertrag.newHausratZusatzdeckung(fahrraddiebstahlTyp);

    assertThat(fahrraddiebstahl.getVersSumme(), is(Money.euro(5_000)));
}

@Test
void testGetVerSumme_Zusatzdeckung_Überspannung_NichtMaximiert() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setVersSumme(Money.euro(600_000));
    HausratZusatzdeckungstyp überspannungTyp = (HausratZusatzdeckungstyp) repository
            .getProductComponent("hausrat.Überspannung 2023-01");
    HausratZusatzdeckung überspannung = hausratVertrag.newHausratZusatzdeckung(überspannungTyp);

    assertThat(überspannung.getVersSumme(), is(Money.euro(30_000)));
}
```

17.9 Übung 13.4: Aufruf von Formeln

17.9.1 Klasse: HausratZusatzdeckung

```
public Money getJahresbasisbeitrag() {
    HausratZusatzdeckungstyp productCmpt = getHausratZusatzdeckungstyp();
    // begin-user-code
    if (productCmpt == null) {
        return null;
    }
    // Belegung der Berechnungsparameter
    HausratZusatzdeckung hausratZusatzdeckung = this;
    return productCmpt.computeJahresbasisbeitrag(hausratZusatzdeckung);
    // end-user-code
}
```

17.9.2 Testklasse HausratProduktTest

```
private static ClassloaderRuntimeRepository repository;
private static HausratProdukt hrKompakt;

@BeforeAll
static void setUp() {
    // Repository erzeugen
    repository = ClassloaderRuntimeRepository
            .create("org/faktorips/schulung/hausratprodukte/internal/faktorips-repository-toc.xml");
    repository.setFormulaEvaluatorFactory(new GroovyFormulaEvaluatorFactory());
    // Referenz auf das Kompaktprodukt aus dem Repository holen
    hrKompakt = (HausratProdukt) repository.getProductComponent("hausrat.HR-Kompakt 2023-01");
}

@Test
void testBerechneJahresbasisbeitrag_Zusatzdeckung_Fahrraddiebstahl() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setVersSumme(Money.euro(600_000));
    HausratZusatzdeckungstyp fahrraddiebstahlTyp = (HausratZusatzdeckungstyp) repository
            .getProductComponent("hausrat.Fahrraddiebstahl 2023-01");
    HausratZusatzdeckung fahrraddiebstahl = hausratVertrag.newHausratZusatzdeckung(fahrraddiebstahlTyp);

    assertThat(fahrraddiebstahl.getJahresbasisbeitrag(), is(Money.euro(500)));
}

@Test
void testBerechneJahresbasisbeitrag_Zusatzdeckung_Überspannung() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setVersSumme(Money.euro(600_000));
    HausratZusatzdeckungstyp überspannungTyp = (HausratZusatzdeckungstyp) repository
            .getProductComponent("hausrat.Überspannung 2023-01");
    HausratZusatzdeckung überspannung = hausratVertrag.newHausratZusatzdeckung(überspannungTyp);

    assertThat(überspannung.getJahresbasisbeitrag(), is(Money.euro(930)));
}
```

17.10 Übung 14.2: Validierung

17.10.1 Klasse: HausratVertrag

```
protected boolean pruefeWohnflaeche(MessageList ml, IValidationContext context) {
    if (!getAllowedValuesForWohnflaeche().contains(getWohnflaeche())) {
        // begin-user-code
        ml.add(createMessageForRulePruefeWohnflaeche(context,
                                                     getAllowedValuesForWohnflaeche(),
                                                     getWohnflaeche()));
        // end-user-code
    }
    return CONTINUE_VALIDATION;
}

private static final Pattern FIVE_DIGITS = Pattern.compile("\\d{5}");
protected boolean pruefePLZ(MessageList ml, IValidationContext context) {
    // begin-user-code
    if (plz != null && !FIVE_DIGITS.matcher(plz).matches()) {
        ml.add(createMessageForRulePruefePlz(context, plz));
    }
    return CONTINUE_VALIDATION;
    // end-user-code
}
```

17.10.2 Testklasse HausratProduktTest

```
@Test
void testPrüfeWohnfläche_OK() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setVersSumme(Money.euro(100_000));

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, is(isEmpty()));
}

@Test
void testPrüfeWohnfläche_ZuHoch() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(250);

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    assertThat(validationMessages, hasMessageCode(HausratVertrag.MSG_CODE_PRUEFE_WOHNFLAECHE));
    Message message = validationMessages.getMessageByCode(HausratVertrag.MSG_CODE_PRUEFE_WOHNFLAECHE);
    assertThat(message, containsText("0-100"));
    assertThat(message, containsText("250"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_WOHNFLAECHE));
}

@Test
void testPrüfeWohnflaeche_NichtGesetzt() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    assertThat(validationMessages, hasMessageCode(HausratVertrag.MSG_CODE_PRUEFE_WOHNFLAECHE));
    Message message = validationMessages.getMessageByCode(HausratVertrag.MSG_CODE_PRUEFE_WOHNFLAECHE);
    assertThat(message, is(notNullValue()));
    assertThat(message, containsText("0-100"));
    assertThat(message, containsText("null"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_WOHNFLAECHE));
}
```

```
@Test
void testPrüfePLZ_OK() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setVersSumme(Money.euro(100_000));
    hausratVertrag.setPlz("50678");

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, is(isEmpty()));
}

@Test
void testPrüfePLZ_ZuKurz() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setPlz("8000");

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    assertThat(validationMessages, hasMessageCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ));
    Message message = validationMessages.getMessageByCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ);
    assertThat(message, containsText("8000"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_PLZ));
}

@Test
void testPrüfePLZ_ZuLang() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setPlz("12345678");

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    assertThat(validationMessages, hasMessageCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ));
    Message message = validationMessages.getMessageByCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ);
    assertThat(message, containsText("12345678"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_PLZ));
}

@Test
void testPrüfePLZ_KeineZiffer() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setPlz("ABCDE");

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    assertThat(validationMessages, hasMessageCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ));
    Message message = validationMessages.getMessageByCode(HausratVertrag.MSG_CODE_PRUEFE_PLZ);
    assertThat(message, containsText("ABCDE"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_PLZ));
}
```

17.11 Übung 14.4 Generische Validierung

17.11.1 Testklasse HausratProduktTest

```
@Test
void testGenerischeValidierung_OK() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setPlz("50678");
    hausratVertrag.setVersSumme(Money.euro(100_000));

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, is(isEmpty()));
}

@Test
void testGenerischeValidierung_VersSummeZuHoch() {
    HausratVertrag hausratVertrag = hrKompakt.createHausratVertrag();
    hausratVertrag.setWohnflaeche(75);
    hausratVertrag.setPlz("50678");
    hausratVertrag.setVersSumme(Money.euro(100_000_000));

    MessageList validationMessages = hausratVertrag.validate(new ValidationContext());

    assertThat(validationMessages, containsErrorMessage());
    String messageCode = GenericRelevanceValidation.Error.ValueNotInValueSet.getDefaultMessageCode(
        HausratVertrag.class,
        HausratVertrag.PROPERTY_VERSSUMME);
    assertThat(validationMessages, hasMessageCode(messageCode));
    Message message = validationMessages.getMessageByCode(messageCode);
    assertThat(message, containsText("Versicherungssumme"));
    assertThat(message, containsText("10000.00 EUR"));
    assertThat(message, containsText("2000000.00 EUR"));
    assertThat(message, hasInvalidObject(hausratVertrag, HausratVertrag.PROPERTY_VERSSUMME));
}
```

17.12 Beispiel: JUnit-Testfall mit InMemoryRepository

```
class HausratTest {

    private InMemoryRuntimeRepository repository = new InMemoryRuntimeRepository();

    /**
     * Diese Methode testet die Beitragsberechnung der Grunddeckung isoliert, also
     * ohne auf produktive Produktdaten zuzugreifen
     */
    @Test
    void testBerechneJahresbasisbeitrag() {

        createTestContent();

        // Das TestProdukt aus dem Repository abfragen
        HausratProdukt testProdukt = (HausratProdukt) repository.getProductComponent("TestProdukt 2023-
01");

        // Erzeugen eines hausratvertrags mit der Factorymethode des Produktes
        HausratVertrag vertrag = testProdukt.createHausratVertrag();

        // Vertragsattribute setzen
        vertrag.setVersSumme(Money.euro(100000));
        vertrag.setPlz("30000"); // Tarifzone III

        // Grunddeckungsbaustein holen, der dem Produkt in der Generation zugeordnet
        // ist.
        HausratGrunddeckungstyp grunddeckungsbaustein =
vertrag.getHausratProdukt().getHausratGrunddeckungstyp();

        // Grunddeckung erzeugen und zum Vertag hinzufuegen
        HausratGrunddeckung deckung = vertrag.newHausratGrunddeckung(grunddeckungsbaustein);

        deckung.berechneJahresbasisbeitrag();
        assertThat(deckung.getJahresbasisbeitrag(), is(Money.euro(100)));
    }

    private void createTestContent() {
        // Erstellen eines Test-Produktbausteins
        HausratProdukt testProdukt = new HausratProdukt(repository, "TestProdukt 2023-01", "TestProdukt",
                "2023-01");
        testProdukt.setProduktname("TestProdukt");
        testProdukt.setVorschlagVersSummeProQm(Money.euro(500));
        // Hinzufügen des Produktbausteins zum Repository.
        repository.putProductComponent(testProdukt);
        // Erstellen eines Test-Grunddeckungsbausteins
        HausratGrunddeckungstyp testDeckungstyp = new HausratGrunddeckungstyp(repository,
                "TestDeckung 2023-01", "TestDeckung", "2023-01");
        // Der Tabellenname muss der gleiche sein unter dem man die
        // Testtariftabelle im Repository abspeichert. Siehe unten
        testDeckungstyp.setTariftabelleName("Tariftabelle");
        testProdukt.setHausratGrunddeckungstyp(testDeckungstyp);
        // Hinzufügen des Produktbausteins zum Repository.
        repository.putProductComponent(testDeckungstyp);
        // Hinzufügen der Testtarifzonentabelle zum Repository
        repository.putTable(new TarifzonentabelleHausrat(List.of(
                new TarifzonentabelleRow("10000", "10001", "I"),
                new TarifzonentabelleRow("20000", "20002", "II"),
                new TarifzonentabelleRow("30000", "30003", "III"))));

        // Hinzufügen der Testtariftabelle zum Repository. Wichtig ist, da es
        // eine Tabelle mit multiplen Inhalten ist, muss der Name der Tabelle
        // angegeben werden, unter dem die Tabelle im Repository gespeichert
        // wird.
        repository.putTable(new TariftabelleHausrat(List.of(
                new TariftabelleHausratRow("I", Decimal.valueOf("0.6")),
                new TariftabelleHausratRow("II", Decimal.valueOf("0.8")),
                new TariftabelleHausratRow("III", Decimal.valueOf("1.0")))),
                "Tariftabelle");
    }
}
```

17.13 Beispiel IpsTest

17.13.1 Klasse BerechnungsTest

```
@Override
public void setRepository(IRuntimeRepository runtimeRepository) {
    ((ClassloaderRuntimeRepository) runtimeRepository)
            .setFormulaEvaluatorFactory(new GroovyFormulaEvaluatorFactory());
    super.setRepository(runtimeRepository);
}

public void executeBusinessLogic() {
    // begin-user-code
    inputHausratVertrag.getHausratGrunddeckung().berechneJahresbasisbeitrag();
    // end-user-code
}

public void executeAsserts(IpsTestResult result) {
    // begin-user-code
    assertEquals(getExtensionAttributeValue(erwartetHausratVertrag,
TESTATTR_HAUSRAT_VERTRAG_VORSCHLAG_VERS_SUMME),
            inputHausratVertrag.getVorschlagVersSumme(),
            result, "HausratVertrag#0", TESTATTR_HAUSRAT_VERTRAG_VORSCHLAG_VERS_SUMME);
    assertEquals(getExtensionAttributeValue(erwartetHausratVertrag, TESTATTR_HAUSRAT_VERTRAG_TARIFZONE),
            inputHausratVertrag.getTarifzone(),
            result, "HausratVertrag#0", TESTATTR_HAUSRAT_VERTRAG_TARIFZONE);

    HausratGrunddeckung erwartetGrunddeckung = erwartetHausratVertrag.getHausratGrunddeckung();
    HausratGrunddeckung inputGrunddeckung = inputHausratVertrag.getHausratGrunddeckung();
    assertEquals(erwartetGrunddeckung.getJahresbasisbeitrag(),
            inputGrunddeckung.getJahresbasisbeitrag(),
            result, "HausratVertrag#0.HausratGrunddeckung#0",
HausratGrunddeckung.PROPERTY_JAHRESBASISBEITRAG);
    assertEquals(
            getExtensionAttributeValue(erwartetGrunddeckung,
                    TESTATTR_HAUSRAT_GRUNDDECKUNG_BEITRAG_GEMAESS_ZAHLWEISE),
            inputGrunddeckung.getBeitragGemaessZahlweise(),
            result, "HausratVertrag#0.HausratGrunddeckung#0",
            TESTATTR_HAUSRAT_GRUNDDECKUNG_BEITRAG_GEMAESS_ZAHLWEISE);

    for (int i = 0; i < erwartetHausratVertrag.getNumOfHausratZusatzdeckungen(); i++) {
        HausratZusatzdeckung erwartetZusatzdeckung = erwartetHausratVertrag.getHausratZusatzdeckung(i);
        HausratZusatzdeckung inputZusatzdeckung = inputHausratVertrag.getHausratZusatzdeckung(i);
        assertEquals(getExtensionAttributeValue(erwartetZusatzdeckung,
TESTATTR_HAUSRAT_ZUSATZDECKUNG_VERS_SUMME),
                inputZusatzdeckung.getVersSumme(),
                result, "HausratVertrag#0.HausratZusatzdeckung#" + i,
TESTATTR_HAUSRAT_ZUSATZDECKUNG_VERS_SUMME);
        assertEquals(
                getExtensionAttributeValue(erwartetZusatzdeckung,
                        TESTATTR_HAUSRAT_ZUSATZDECKUNG_JAHRESBASISBEITRAG),
                inputZusatzdeckung.getJahresbasisbeitrag(),
                result, "HausratVertrag#0.HausratZusatzdeckung#" + i,
                TESTATTR_HAUSRAT_ZUSATZDECKUNG_JAHRESBASISBEITRAG);
    }
    // end-user-code
}

public class HausratProductTestRunner extends IpsTestSuiteJUnit5Adapter {
    @TestFactory
    public Stream<DynamicTest> getTests() {
        IRuntimeRepository repository = ClassloaderRuntimeRepository.create(
                "org/faktorips/schulung/hausratprodukte/internal/faktorips-repository-toc.xml");
        return createTests(repository.getIpsTest(""));
    }
}
```